

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 4**



VIEWMODEL AND DEBUGGING

Oleh:

Ririn Citra Lestari

NIM. 2310817120012

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025**

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 4

Laporan Praktikum Pemrograman Mobile Modul 4: ViewModel and Debugging ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Ririn Citra Lestari
NIM : 2310817220018

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar
NIM. 2210817210012

Andreyan Rizky Baskara, S.Kom., M.Kom.
NIP. 19930703 201903 01 011

DAFTAR ISI

LEMBAR PENGESAHAN.....	2
DAFTAR ISI.....	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL.....	5
SOAL 1.....	6
A. SOURCE CODE.....	6
B. OUTPUT PROGRAM.....	20
C. PEMBAHASAN	24
D. TAUTAN GIT.....	26
SOAL 2.....	27

DAFTAR GAMBAR

Gambar 1. Contoh Penggunaan Debugger.....	6
Gambar 2. Breakpoint (Compose)	21
Gambar 3. Step Into (F7) Masuk Ke Fungsi (Compose)	21
Gambar 4. Step Over (F8) Digunakan (Compose).....	21
Gambar 5. Step Out (Shift + F8) Keluar Dari Fungsi (Compose)	22
Gambar 6. Resume (F9) Untuk Melanjutkan Eksekusi (Compose).....	22
Gambar 7. Breakpoint (XML).....	22
Gambar 8. Debug (XML).....	23
Gambar 9. Step Into (F7) Masuk Ke Fungsi (XML).....	23
Gambar 10. Step Over (F8) Digunakan (XML).....	23
Gambar 11. Step Out (Shift + F8) Keluar Dari Fungsi (XML).....	24

DAFTAR TABEL

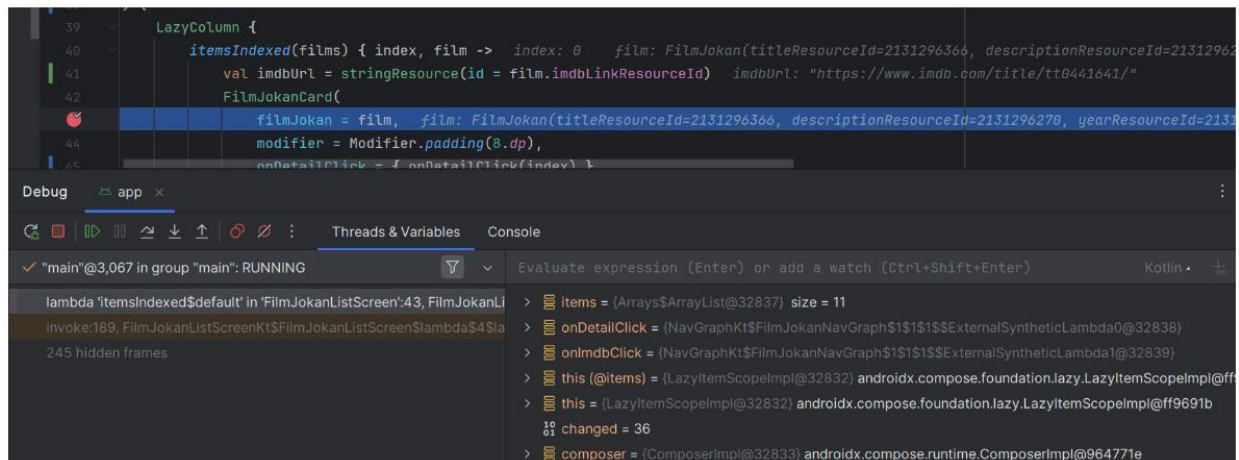
Tabel 1. Source Code Jawaban Soal 1	6
Tabel 2. Source Code Jawaban Soal 1	7
Tabel 3. Source Code Jawaban Soal 1	8
Tabel 4. Source Code Jawaban Soal 1	8
Tabel 5. Source Code Jawaban Soal 1	9
Tabel 6. Source Code Jawaban Soal 1	11
Tabel 7. Source Code Jawaban Soal 1	12
Tabel 8. Source Code Jawaban Soal 1	13
Tabel 9. Source Code Jawaban Soal 1	13
Tabel 10. Source Code Jawaban Soal 1	13
Tabel 11. Source Code Jawaban Soal 1	14
Tabel 12. Source Code Jawaban Soal 1	14
Tabel 13. Source Code Jawaban Soal 1	16
Tabel 14. Source Code Jawaban Soal 1	17
Tabel 15. Source Code Jawaban Soal 1	17
Tabel 16. Source Code Jawaban Soal 1	19
Tabel 17. Source Code Jawaban Soal 1	20
Tabel 18. Source Code Jawaban Soal 1	20

SOAL 1

Soal Praktikum:

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

- Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
- Gunakan ViewModelFactory untuk membuat parameter dengan tipe data String di dalam ViewModel
- Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
- Install dan gunakan library Timber untuk logging event berikut:
 - Log saat data item masuk ke dalam list
 - Log saat tombol Detail dan tombol Explicit Intent ditekan
 - Log data dari list yang dipilih ketika berpindah ke halaman Detail
- Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out



Gambar 1. Contoh Penggunaan Debugger

A. SOURCE CODE

Compose:

MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1

```
package com.example.listfilm

import android.os.Bundle
import androidx.activity.ComponentActivity
```

```

import androidx.activity.compose.setContent
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.compose.rememberNavController
import com.example.listfilm.ui.screen.AppNavGraph
import com.example.listfilm.viewmodel.FilmViewModel
import com.example.listfilm.viewmodel.FilmViewModelFactory
import timber.log.Timber

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        Timber.d("MainActivity onCreate() called")

        setContent {
            val navController = rememberNavController()
            val viewModel: FilmViewModel = viewModel(
                factory = FilmViewModelFactory("Local Data")
            )
            AppNavGraph(navController = navController, viewModel =
viewModel)
        }
    }
}

```

FilmData.kt

Tabel 2. Source Code Jawaban Soal 1

```

package com.example.listfilm.data

import com.example.listfilm.R

val filmList = listOf(
    Film(
        title = "Happiness",
        description = "Drama Korea Happiness adalah drama
Korea yang mengusung tema tentang perjuangan manusia
menghadapi penyakit menular di dunia yang terjebak dalam
perbedaan kelas sosial.",
        imageResId = R.drawable.happiness,
        url =
"https://www.bilibili.tv/id/video/4792916821151744",
        genre = "Action, Thriller, Drama, Sci-Fi",
        year = 2021
    ),
    Film(
        title = "All Of Us Are Dead",
        description = "Sebuah SMA menjadi titik nol merebaknya
wabah virus zombi. Para murid yang terperangkap pun harus
berjuang untuk kabur jika tak mau terinfeksi dan berubah
menjadi buas.",
        imageResId = R.drawable.aouad,
        url = "https://www.bilibili.tv/id/video/2049830922",
        genre = "Action, Thriller, Horror, Sci-Fi",
        year = 2022
    ),
)

```

```

    Film(
        title = "The Trauma Code",
        description = "Seorang ahli bedah jenius yang tergabung
dengan tim trauma parah memimpin sebuah tim untuk berjuang di sebuah
rumah sakit universitas. ",
        imageResId = R.drawable.traumacenter,
        url = "https://www.bilibili.tv/id/video/4794455678652416",
        genre = "Action, Drama, Medical, Fantasy",
        year = 2025
    ),
    Film(
        title = "Newtopia",
        description = "sebuah drama yang mengisahkan perjalanan
seorang tentara bernama Jae Yoon. Tanpa diduga, ia harus menghadapi
wabah zombie bersama mantan kekasihnya, Kang Yeong Ju.",
        imageResId = R.drawable.newtopia,
        url = "https://www.bilibili.tv/id/video/4794465462128640",
        genre = "Action, Thriller, Romance, Fantasy",
        year = 2025
    ),
    Film(
        title = "Snow Drop",
        description = "Drama Snowdrop ini berlatar Korea Selatan pada
tahun 1987, menceritakan tentang Im Soo Ho (Jung Hae In) seorang
mahasiswa di sebuah universitas bergengsi yang tiba-tiba masuk ke
asrama wanita dengan berlumuran darah. Disana ia bertemu dengan
seorang mahasiswi Eun Young Ro (Kim Ji Soo) yang ceria dan
menyenangkan. Young Ro lantas menyembunyikan Soo Ho dan merawat
lukanya bahkan disaat pengawasan ketat.",
        imageResId = R.drawable.snowdrop,
        url = "https://www.bilibili.tv/id/video/2009361160",
        genre = "Romance, Drama, Melodrama, Political",
        year = 2021
    )
)

```

Film.kt

Tabel 3. Source Code Jawaban Soal 1

```

package com.example.listfilm.data

data class Film(
    val title: String,
    val year: Int,
    val genre: String,
    val description: String,
    val imageResId: Int,
    val url: String
)

```

FilmDetailScreen.kt

Tabel 4. Source Code Jawaban Soal 1


```
package com.example.listfilm.ui.screen

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
```

```

import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.listfilm.data.Film

@Composable
fun FilmDetailScreen(film: Film) {
    Column(modifier = Modifier.padding(16.dp)) {
        Image(
            painter = painterResource(id = film.imageResId),
            contentDescription = null,
            modifier = Modifier
                .fillMaxWidth()
                .height(200.dp)
                .clip(RoundedCornerShape(12.dp)),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = film.title, style =
MaterialTheme.typography.headlineSmall)
        Text(text = "${film.genre} • ${film.year}")
        Spacer(modifier = Modifier.height(8.dp))
        Text(text = film.description)
    }
}

```

FilmListScreen.kt

Tabel 5. Source Code Jawaban Soal 1

```

package com.example.listfilm.ui.screen

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier

```

```

import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.core.net.toUri
import androidx.navigation.NavHostController
import com.example.listfilm.viewmodel.FilmViewModel

@Composable
fun FilmListScreen(navController: NavHostController, viewModel:
FilmViewModel) {
    val context = LocalContext.current
    val films = viewModel.films.collectAsState().value

    LazyColumn(contentPadding = PaddingValues(16.dp)) {
        items(films) { film ->
            Card(
                shape = RoundedCornerShape(16.dp),
                modifier = Modifier
                    .padding(bottom = 16.dp)
                    .fillMaxWidth()
            ) {
                Column(Modifier.padding(16.dp)) {
                    Image(
                        painter = painterResource(id =
film.imageResId),
                        contentDescription = null,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp)
                            .clip(RoundedCornerShape(12.dp)),
                        contentScale = ContentScale.Crop
                    )
                    Spacer(Modifier.height(8.dp))
                    Row(
                        horizontalArrangement =
Arrangement.SpaceBetween,
                        modifier = Modifier.fillMaxWidth()
                    ) {
                        Text(text = film.title, fontWeight =
FontWeight.Bold)
                        Text(text = "${film.year}")
                    }
                    Row(
                        horizontalArrangement =
Arrangement.SpaceBetween,
                        modifier = Modifier.fillMaxWidth()
                    ) {
                        Text(text = film.genre)
                        Text(text = "Korean")
                    }
                }
            }
        }
    }
}

```

```

    }
    Spacer(Modifier.height(8.dp))
    Row(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        Button(onClick = {
            viewModel.logOpenSiteClicked(film)
            val intent = Intent(Intent.ACTION_VIEW,
                film.url.toUri())
            context.startActivity(intent)
        }) {
            Text("Open Site")
        }
        Button(onClick = {
            viewModel.logDetailClicked(film)
            navController.navigate("detail/${Uri.encode(film.title)}")
        }) {
            Text("Detail")
        }
    }
}
}
}
}

```

Tabel 6. Source Code Jawaban Soal 1

```
package com.example.listfilm.ui.screen

import android.net.Uri
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import com.example.listfilm.viewmodel.FilmViewModel

@Composable
fun AppNavGraph(navController: NavHostController, viewModel: FilmViewModel) {
    val films = viewModel.films.collectAsState().value
    NavHost(navController = navController, startDestination = "list")
    {
        composable("list") {

```

```

        FilmListScreen(navController = navController, viewModel =
viewModel)
    }
    composable("detail/{title}") { backStackEntry ->
        val rawTitle = backStackEntry.arguments?.getString("title")
        val decodedTitle = rawTitle?.let { Uri.decode(it) }
        val film = films.find { it.title == decodedTitle }
        if (film != null) {
            FilmDetailScreen(film = film)
        } else {
            Text("Film not found", modifier =
Modifier.fillMaxSize())
        }
    }
}

```

FilmViewModel.kt

Tabel 7. Source Code Jawaban Soal 1

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.listfilm.data.Film
import com.example.listfilm.data.filmList
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import timber.log.Timber

class FilmViewModel(private val source: String) : ViewModel() {
    private val _films = MutableStateFlow<List<Film>>(emptyList())
    val films: StateFlow<List<Film>> = _films

    init {
        viewModelScope.launch {
            _films.value = filmList
            Timber.d("Film list loaded from $source with
${filmList.size} items")
        }
    }

    fun logDetailClicked(film: Film) {
        Timber.d("Detail clicked: ${film.title}")
    }

    fun logOpenSiteClicked(film: Film) {
        Timber.d("Open site clicked: ${film.url}")
    }
}

```

FilmViewModelFactory.kt

Tabel 8. Source Code Jawaban Soal 1

```
package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class FilmViewModelFactory(private val source: String) :
    ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(FilmViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return FilmViewModel(source) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

ListFilmApp.kt

Tabel 9. Source Code Jawaban Soal 1

```
package com.example.listfilm

import android.app.Application
import timber.log.Timber

class ListFilmApp : Application() {
    override fun onCreate() {
        super.onCreate()
        Timber.plant(Timber.DebugTree())
    }
}
```

XML:

Activity_main.xml

Tabel 10. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvFilm"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:padding="16dp"/>
    </LinearLayout>
```

activity_detail.xml

Tabel 11. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/imgDetail"
            android:layout_width="match_parent"
            android:layout_height="351dp"
            android:scaleType="centerCrop" />

        <TextView
            android:id="@+id/tvDetailTitle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:layout_marginTop="16dp" />

        <TextView
            android:id="@+id/tvDetailGenre"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="14sp"
            android:textColor="#666"/>

        <TextView
            android:id="@+id/tvDetailDescription"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp" />
    </LinearLayout>
</ScrollView>
```

item_film.xml

Tabel 12. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto">
```

```

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="wrap_content"
card_view:cardCornerRadius="12dp"
card_view:cardElevation="4dp"
android:layout_marginBottom="16dp"
tools:ignore="MissingClass">

<LinearLayout
    android:orientation="vertical"
    android:padding="16dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/imgFilm"
        android:layout_width="match_parent"
        android:layout_height="259dp"
        android:adjustViewBounds="true"
        android:scaleType="centerCrop" />

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textSize="16sp"
        android:layout_marginTop="8dp"/>

    <TextView
        android:id="@+id/tvInfo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="#666"/>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:gravity="end">

        <Button
            android:id="@+id/btnOpenSite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Open Site"/>

        <Button
            android:id="@+id/btnDetail"
            android:layout_width="wrap_content"

```



```

        android:layout_height="wrap_content"
        android:text="Detail"
        android:layout_marginStart="8dp"/>
    </LinearLayout>

</LinearLayout>
</androidx.cardview.widget.CardView>

```

DetailActivity.kt

Tabel 13. Source Code Jawaban Soal 1

```

package com.example.listfilm

import android.annotation.SuppressLint
import android.os.Bundle
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.listfilm.data.filmList

class DetailActivity : AppCompatActivity() {

    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_detail)

        val title = intent.getStringExtra("film")
        if (title == null) {
            Toast.makeText(this, "Data film tidak diterima",
                Toast.LENGTH_SHORT).show()
            finish()
            return
        }
    }
}

```

```

        val film = filmList.find { it.title == title }

        if (film == null) {

            Toast.makeText(this, "Film tidak ditemukan",
Toast.LENGTH_SHORT).show()

            finish()

            return

        }

findViewById<ImageView>(R.id.imgDetail).setImageResource(film.imageResId)

        findViewById<TextView>(R.id.tvDetailTitle).text = film.title

        findViewById<TextView>(R.id.tvDetailGenre).text = "${film.genre}
• ${film.year}"

        findViewById<TextView>(R.id.tvDetailDescription).text =
film.description

    }

}

```

Film.kt

Tabel 14. Source Code Jawaban Soal 1

```

package com.example.listfilm.data

data class Film(
    val title: String,
    val year: Int,
    val genre: String,
    val description: String,
    val imageResId: Int,
    val url: String
)

```

FilmAdapter.kt

Tabel 15. Source Code Jawaban Soal 1

```

package com.example.listfilm

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.*
import androidx.recyclerview.widget.RecyclerView
import com.example.listfilm.data.Film

```

```

import com.example.listfilm.data.filmList

class FilmAdapter(
    private val context: Context,
    private var listFilm: List<Film>,
    private val onDetailClick: (Film) -> Unit
) : RecyclerView.Adapter<FilmAdapter.FilmViewHolder>() {

    inner class FilmViewHolder(view: View) :
        RecyclerView.ViewHolder(view) {
        val imgFilm: ImageView = view.findViewById(R.id.imgFilm)
        val tvTitle: TextView = view.findViewById(R.id.tvTitle)
        val tvInfo: TextView = view.findViewById(R.id.tvInfo)
        val btnOpenSite: Button = view.findViewById(R.id.btnOpenSite)
        val btnDetail: Button = view.findViewById(R.id.btnDetail)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
        FilmViewHolder {
        val view =
            LayoutInflater.from(parent.context).inflate(R.layout.item_film, parent,
                false)
        return FilmViewHolder(view)
    }

    override fun getItemCount(): Int = listFilm.size

    @SuppressWarnings("NotifyDataSetChanged")
    fun updateData(newFilmList: List<Film>) {
        listFilm = newFilmList
        notifyDataSetChanged()
    }

    override fun onBindViewHolder(holder: FilmViewHolder, position: Int)
    {
        val film = listFilm[position]
        holder.imgFilm.setImageResource(film.imageResId)
        holder.tvTitle.text = film.title
        holder.tvInfo.text = "${film.genre} • ${film.year}"

        holder.btnOpenSite.setOnClickListener {
            val intent = Intent(Intent.ACTION_VIEW, Uri.parse(film.url))
            context.startActivity(intent)
        }

        holder.btnDetail.setOnClickListener {
            onDetailClick(film)
        }
    }
}

```

MainActivity.kt

Tabel 16. Source Code Jawaban Soal 1

```
package com.example.listfilm

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.lifecylceScope
import com.example.listfilm.viewmodel.FilmViewModel
import com.example.listfilm.viewmodel.FilmViewModelFactory
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch
import timber.log.Timber

class MainActivity : AppCompatActivity() {

    private lateinit var rvFilm: RecyclerView
    private lateinit var viewModel: FilmViewModel
    private lateinit var adapter: FilmAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        Timber.plant(Timber.DebugTree())

        setContentView(R.layout.activity_main)

        // Setup RecyclerView
        rvFilm = findViewById(R.id.rvFilm)
        rvFilm.layoutManager = LinearLayoutManager(this)

        // Setup ViewModel dengan factory
        val factory = FilmViewModelFactory("Devi")
        viewModel = ViewModelProvider(this,
factory) [FilmViewModel::class.java]

        // Setup Adapter dengan callback onClick
        adapter = FilmAdapter(this, emptyList()) { film ->
            Timber.i("Button Detail clicked for: ${film.title}")
            viewModel.selectFilm(film)
            val intent = Intent(this,
DetailActivity::class.java).apply {
                putExtra("film", film.title)
            }
            startActivity(intent)
        }
        rvFilm.adapter = adapter

        lifecycleScope.launch {
            viewModel.films.collectLatest { films ->
```

```

        adapter.updateData(films)
    }
}
}
}

```

FilmViewModel.kt

Tabel 17. Source Code Jawaban Soal 1

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import com.example.listfilm.data.Film
import com.example.listfilm.data.filmList as staticFilmList // alias
    biar aman
import timber.log.Timber

class FilmViewModel(private val username: String) : ViewModel() {

    private val _filmList = MutableStateFlow<List<Film>>(emptyList())
    val films: StateFlow<List<Film>> = _filmList

    private val _selectedFilm = MutableStateFlow<Film?>(null)
    val selectedFilm: StateFlow<Film?> = _selectedFilm

    init {
        Timber.i("[${username}] FilmViewModel initialized.")
        loadFilms()
    }

    private fun loadFilms() {
        _filmList.value = staticFilmList // ini refer ke file data
        Timber.i("[${username}] Film list loaded with
    ${staticFilmList.size} items.")
    }

    fun selectFilm(film: Film) {
        _selectedFilm.value = film
        Timber.i("[${username}] Film selected: ${film.title}")
    }
}

```

FilmViewModelFactory.kt

Tabel 18. Source Code Jawaban Soal 1

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

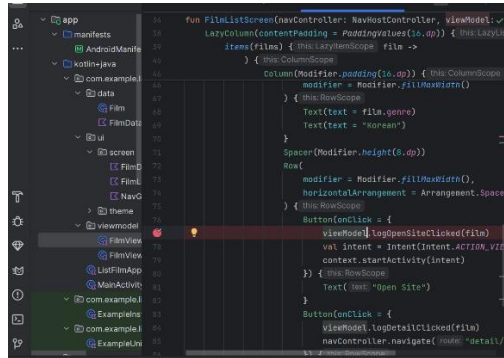
```

```

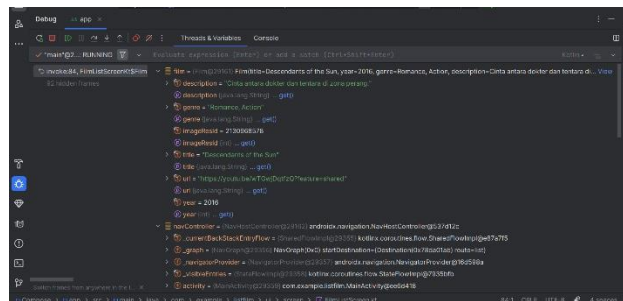
class FilmViewModelFactory(private val username: String) :
    ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(FilmViewModel::class.java)) {
            return FilmViewModel(username) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}

```

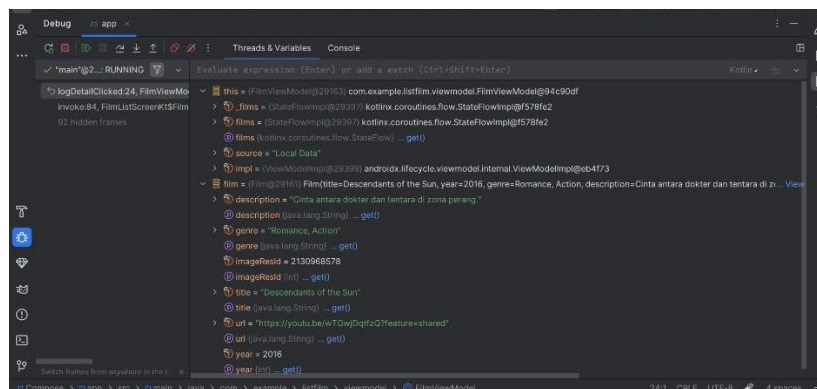
B. OUTPUT PROGRAM



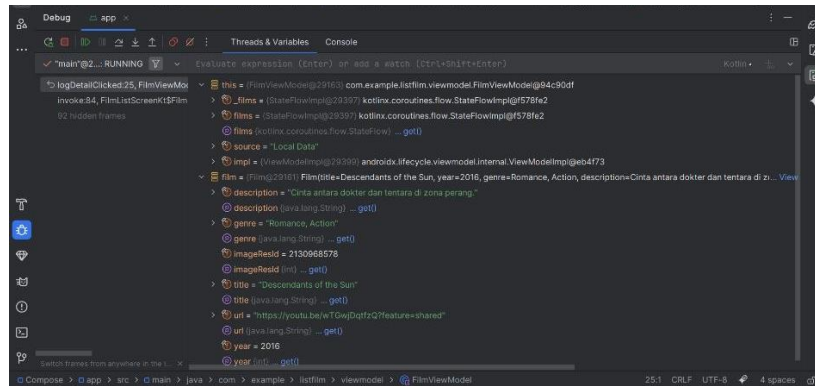
Gambar 2. Breakpoint (Compose)



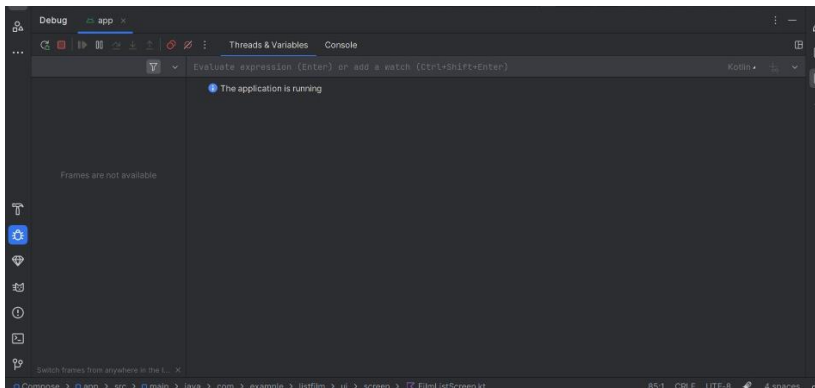
Gambar 3. Step Into (F7) Masuk Ke Fungsi (Compose)



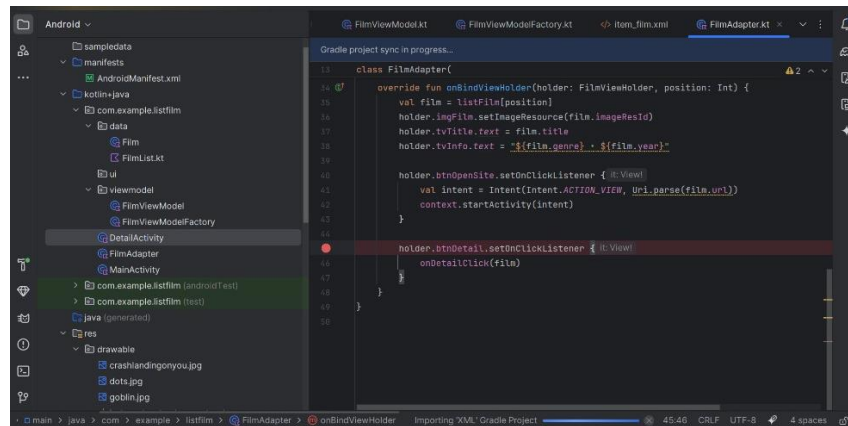
Gambar 4. Step Over (F8) Digunakan (Compose)



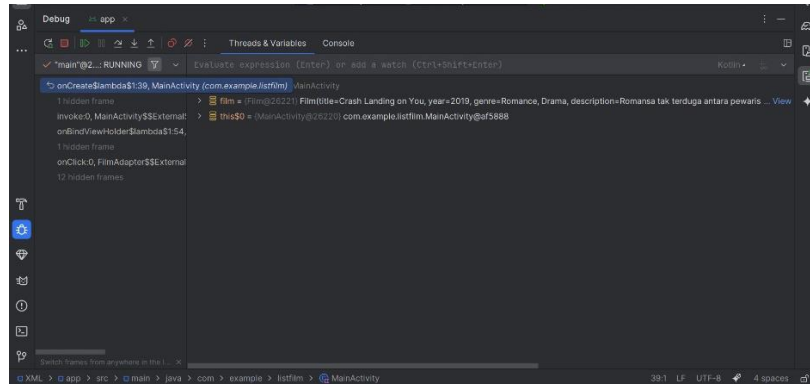
Gambar 5. Step Out (Shift + F8) Keluar Dari Fungsi (Compose)



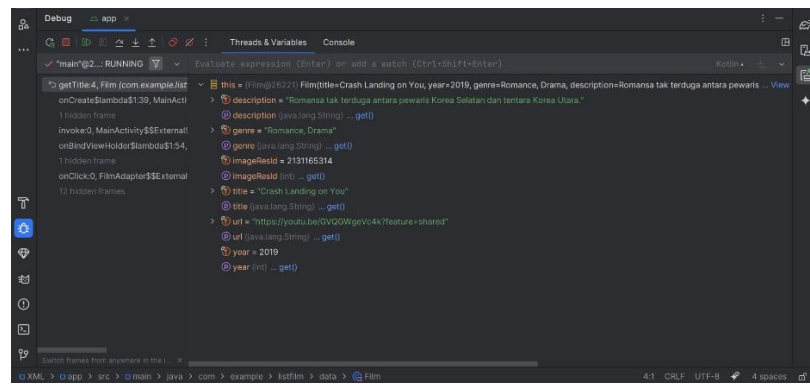
Gambar 6. Resume (F9) Untuk Melanjutkan Eksekusi (Compose)



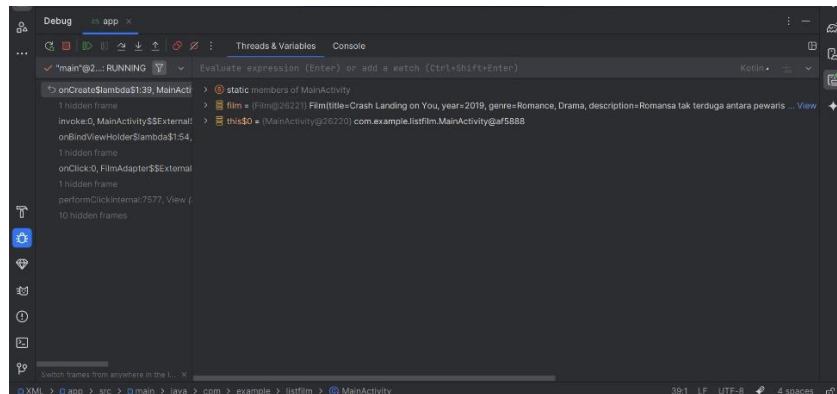
Gambar 7. Breakpoint (XML)



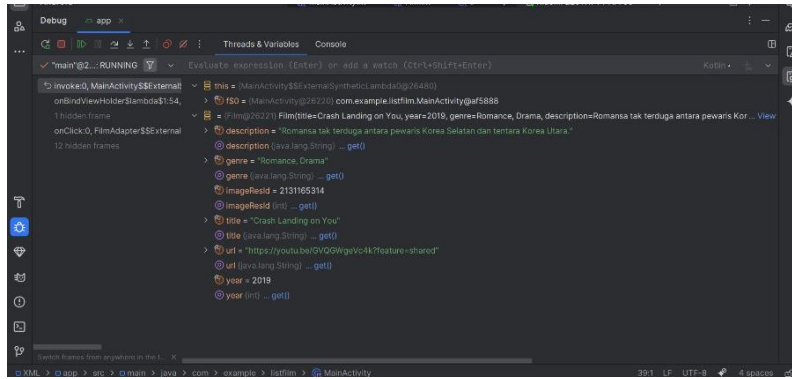
Gambar 8. Debug (XML)



Gambar 9. Step Into (F7) Masuk Ke Fungsi (XML)



Gambar 10. Step Over (F8) Digunakan (XML)



Gambar 11. Step Out (Shift + F8) Keluar Dari Fungsi (XML)

C. PEMBAHASAN

Kode ini lanjutan dari Modul 3 dengan penambahan beberapa modifikasi penting sesuai instruksi. Pertama, untuk memenuhi poin (a), digunakan kelas FilmViewModel sebagai ViewModel yang bertanggung jawab penuh terhadap pengelolaan dan penyimpanan data list item (film). Dengan pendekatan ini, data tidak disimpan di Activity atau Fragment, melainkan dikelola secara terpusat dan reaktif melalui ViewModel, yang memastikan arsitektur aplikasi bersih dan mengikuti prinsip *separation of concerns*.

Pada poin (b), implementasi FilmViewModelFactory digunakan untuk memberikan parameter bertipe String (dalam hal ini, nama pengguna seperti "Devi") ke dalam ViewModel. Factory ini memungkinkan ViewModelProvider membuat instance ViewModel dengan parameter tambahan, karena secara default ViewModel tidak menerima argumen konstruktor.

Untuk poin (c), aplikasi ini menggunakan StateFlow pada FilmViewModel untuk mengelola data daftar film (_filmList) dan film yang dipilih (_selectedFilm). Dengan menggunakan StateFlow, UI (seperti RecyclerView di XML dan LazyColumn di Compose) dapat berlangganan pada data dan secara otomatis *recompose* atau *refresh* ketika ada perubahan.

Selain itu, aksi onClick pada tombol "Detail" juga dikirim ke ViewModel menggunakan fungsi selectFilm(film) yang memperbarui _selectedFilm.

Pada poin (d), logging menggunakan **Timber** telah diimplementasikan. Logging ini dilakukan saat:

- Data film pertama kali dimuat dari filmList (loadFilms()), dengan mencetak jumlah item.
- Tombol "Detail" ditekan pada setiap item (baik di Compose maupun XML/RecyclerView), yang mencetak judul film yang diklik.
- Saat berpindah ke halaman DetailActivity, data film yang dipilih juga dicatat menggunakan Timber.i.

Untuk poin (e), proses debugging dilakukan menggunakan **Debugger** di Android Studio. Sebuah *breakpoint* yang relevan dapat ditempatkan pada fungsi onBindViewHolder di FilmAdapter, tepat saat tombol "Detail" ditekan. Dari sini, fitur:

- **Step Into** digunakan untuk masuk ke dalam fungsi onDetailClick() untuk melihat bagaimana data dikirim ke Intent.
- **Step Over** digunakan untuk mengeksekusi baris kode saat ini dan melanjutkan ke baris berikutnya tanpa masuk ke dalam fungsi yang dipanggil.
- **Step Out** digunakan untuk keluar dari fungsi yang sedang dieksekusi dan kembali ke fungsi pemanggil sebelumnya.

Debugger sangat bermanfaat dalam mengidentifikasi alur eksekusi kode dan mencari tahu nilai variabel secara langsung saat aplikasi berjalan. Dengan memanfaatkan fitur-fitur tersebut, pengembang dapat melacak perilaku aplikasi dengan lebih akurat, terutama ketika terjadi bug atau error.

D. TAUTAN GIT

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/rc114/PrakMobile.git>

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya!

Dalam arsitektur aplikasi Android, Application class merupakan titik awal global yang dijalankan sebelum komponen lain seperti Activity, Service, atau BroadcastReceiver. Class ini digunakan untuk mengatur konfigurasi atau inisialisasi yang hanya perlu dilakukan sekali seumur hidup aplikasi. Karena hanya dibuat satu kali dan bertahan selama aplikasi berjalan, Application cocok digunakan untuk mengatur kebutuhan yang bersifat global.

Salah satu fungsi utama dari Application class adalah untuk melakukan inisialisasi global. Misalnya, saat menggunakan library seperti Timber untuk logging, inisialisasi dapat dilakukan di dalam metode onCreate() dari class Application, sehingga bisa langsung digunakan di seluruh bagian aplikasi tanpa perlu setup berulang. Selain itu, library seperti Firebase, analytics, Room database, atau library dependency injection seperti Dagger/Hilt biasanya juga diinisialisasi di sini.

Class Application juga menyediakan applicationContext yang bisa diakses dari mana saja. Konteks ini tidak terikat pada lifecycle sebuah Activity, sehingga aman digunakan untuk operasi yang memerlukan konteks aplikasi, seperti akses ke file, shared preferences, atau inisialisasi database.

Dalam arsitektur modern seperti MVVM yang menggunakan Jetpack Compose, class Application sering digunakan untuk menyiapkan dependency seperti ViewModel, repository, atau singleton database yang dibagikan ke seluruh komponen. Ketika menggunakan dependency injection seperti Hilt, kita juga perlu menandai class turunan Application dengan anotasi @HiltAndroidApp agar proses injeksi dapat berjalan.

Meskipun powerful, penggunaan class Application juga harus hati-hati. Sebaiknya hindari menyimpan reference ke Activity, Context, atau View di dalamnya karena dapat menyebabkan memory leak. Selain itu, jangan meletakkan terlalu banyak logic dalam class ini. Fokus utamanya adalah untuk setup awal yang ringan dan penting bagi seluruh aplikasi.

Secara keseluruhan, Application class berfungsi sebagai pusat konfigurasi awal dan manajemen state global yang dibutuhkan aplikasi selama runtime. Penggunaan yang tepat akan membuat arsitektur aplikasi lebih terstruktur dan efisien.